EZ Tracker

Technical Report

Ryan Maynard, Delroy Christie, Jonas Gamao

April 25, 2019

# 

# Declaration of Joint Authorship

We, Jonas Gamao, Ryan Maynard, Derloy Christie, confirm that this work submitted for assessment is our own, and is expressed in our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. Individual contribution per group member is indicated in the Work Breakdown and Requirements section of this report. A list of the references used is included.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jonas Gamao

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ryan Maynard

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Delroy Christie

# Copy of the Approved Proposal

# Abstract

With today’s society being more concerned about physical health, it becomes increasingly important for technology to address these needs. Our fitness tracker pilot project aims to address this issue particularly for a specific demographic. Presently, the majority of fitness trackers are aimed toward a younger, active and technologically literate group. While effective, they fail to address the needs of our target population by overloading them with information, and complex interfaces. The purpose of this report will explain the details and methods that have gone into the development and creation of a pedometer based project. Consisting of three sensors – a capacitive touch, an accelerometer and magnetometer and OLED display - the EZ Tracker creates entry level health monitoring and statistics at a glance. By taking users physical data (i.e. weight, height, and age), combined with their steps determined by the accelerometer, the device can calculate and display an approximate caloric loss on the native Android application, as well as the OLED display. By combining these technologies and collaborating amongst the members at JRD Developers, EZ Tracker aims to help promote a robust, inexpensive, and simple to use platform for its users.

Table of Contents

[Declaration of Joint Authorship 1](#_Toc6948998)

[Copy of the Approved Proposal 2](#_Toc6948999)

[Abstract 3](#_Toc6949000)

[Table of Contents 4](#_Toc6949001)

[List of Illustrations or Diagrams 7](#_Toc6949002)

[Introduction 8](#_Toc6949003)

[Work Breakdown and Requirements 10](#_Toc6949004)

[Software specifications 11](#_Toc6949005)

[Application Specifications and Breakdown 11](#_Toc6949006)

[Database Specifications and Breakdown 11](#_Toc6949007)

[Hardware Specifications and Breakdown 12](#_Toc6949008)

[Build Instructions 13](#_Toc6949009)

[Budget and Parts 13](#_Toc6949010)

[Assembly 14](#_Toc6949011)

[Preparing the Pi 14](#_Toc6949012)

[Breadboarding and prototyping 18](#_Toc6949013)

[PCB Design 20](#_Toc6949014)

[Soldering 22](#_Toc6949015)

[Power Up 25](#_Toc6949016)

[Unit Testing – LSM303 26](#_Toc6949017)

[Unit Testing – MPR121 27](#_Toc6949018)

[Unit Testing – 128x64 Monochrome LED 28](#_Toc6949019)

[Production Testing 28](#_Toc6949020)

[Overall Description 30](#_Toc6949021)

[Product Functions 30](#_Toc6949022)

[User Classes and Characteristics 33](#_Toc6949023)

[Operating Environment 33](#_Toc6949024)

[Design and Implementation Constraints 34](#_Toc6949025)

[User Documentation 34](#_Toc6949026)

[Assumptions and Dependencies 34](#_Toc6949027)

[Hardware Interfaces 35](#_Toc6949028)

[Software Interfaces 35](#_Toc6949029)

[Communications Interfaces 36](#_Toc6949030)

[System Features 37](#_Toc6949031)

[Simple Mode 37](#_Toc6949032)

[Description and Priority 37](#_Toc6949033)

[Stimulus/Response Sequences 37](#_Toc6949034)

[Functional Requirements 37](#_Toc6949035)

[Other Non-functional Requirements 38](#_Toc6949036)

[Performance Requirements 38](#_Toc6949037)

[Safety Requirements 38](#_Toc6949038)

[Database corruption 38](#_Toc6949039)

[Security Requirements 38](#_Toc6949040)

[Software Quality Attributes 39](#_Toc6949041)

[Business Rules 39](#_Toc6949042)

[Status Reports 40](#_Toc6949043)

[Status Report 1 40](#_Toc6949044)

[Status Report 2 42](#_Toc6949045)

[Status Report 3 44](#_Toc6949046)

[Status Report 4 46](#_Toc6949047)

[Status Report 5 48](#_Toc6949048)

[Conclusions 50](#_Toc6949049)

[Recommendations 51](#_Toc6949050)

[Bibliography 53](#_Toc6949051)

[Appendices 54](#_Toc6949052)

[Appendix A: Glossary 54](#_Toc6949053)

# List of Illustrations or Diagrams

[Figure 1 10](#_Toc6948982)

[Figure 2 13](#_Toc6948983)

[Figure 3 19](#_Toc6948984)

[Figure 4 20](#_Toc6948985)

[Figure 5 21](#_Toc6948986)

[Figure 6 22](#_Toc6948987)

[Figure 7 23](#_Toc6948988)

[Figure 8 23](#_Toc6948989)

[Figure 9 24](#_Toc6948990)

[Figure 10 25](#_Toc6948991)

[Figure 11 25](#_Toc6948992)

[Figure 12 26](#_Toc6948993)

[Figure 13 27](#_Toc6948994)

[Figure 14 28](#_Toc6948995)

[Figure 15 29](#_Toc6948996)

[Figure 16 32](#_Toc6948997)

# Introduction

This report describes the Internet of Things project that focusses on a monitoring system that tracks users’ steps over a predetermined period. The device described in the project is designed to cater for a segment of the market that is often overlooked. It seeks to address the challenges the elderly and technologically inexperienced have with using and navigating complicated applications to do simple tasks. The objective of this undertaking is to simplify the process of using a device to aid the target demographic in their pursuit of better health. The product aims to reduce the complexities in navigating a device to obtain specific information. The introduction of this product is significant because it caters to a segment of the market that no other product addresses.

The information presented will be centered on how the application will be integrated with designated hardware, and how it will allow users to keep track of their daily and weekly calorie and stepping progress. It also presents information about the types of hardware used, the database for storing and retrieving data, and the application used for the manipulation of these. During the building of the Android application, difficulties arose in determining the information being using for the core operation of the project, as well as deciding the relevant information to display. This resulted in a previous version not meeting the scope of our goal, and thus being omitted.

In general, most pedometers use a multitude of sensors for their purpose, especially something as accurate as GPS. EZ Tracker is unique in the sense where it will only have one sensor which will be gauging steps – an accelerometer. Although maybe not as precise as those readily available on the market, it is a unique approach to the idea that simplicity can achieve a relatively complex task.

# Work Breakdown and Requirements

A breakdown of our work schedule

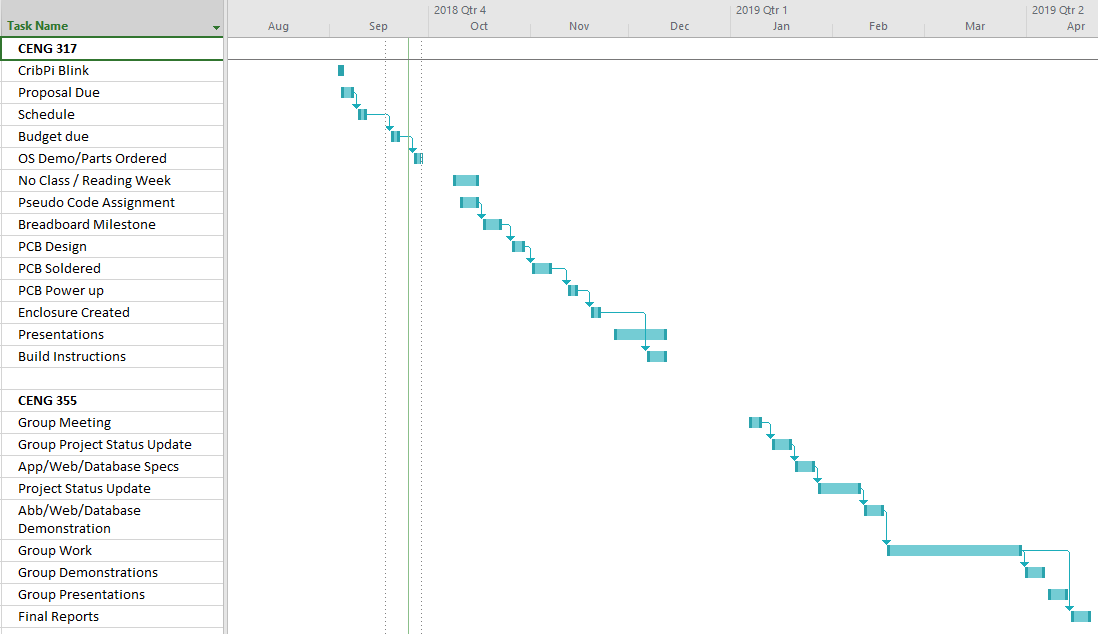


Figure 1

This project took us approximately fifteen weeks to complete, having all of the parts in hand and following the included build instruction, it should take a person approximately one weekend to complete, provided he has a reasonable grasp of basic electronics and programming.

## Software specifications

### Application Specifications and Breakdown

Ryan will be responsible for refining and updating the current version of the Android application. He will update, modify, and adhere to new ideas of the project based on weekly meetings and discussions. Extensive testing will be performed on a number of devices owned by members of the group. Both Delroy and Ryan will jointly be responsible for creating a basic website that will allow the user to login to their created account and using the device, hosted on an apache webserver locally on the Raspberry Pi. They will also be responsible for refining the site for smooth operation and potentially advanced data and metrics which will provide further detail not available on the Android application regarding statistics and hardware data, if possible.

### 

### Database Specifications and Breakdown

The application will be collecting the data and send the results to Google’s Firebase. While the person is using the EZ Tracker hardware, it will need to be connected to the internet. After doing extensive research on various ways of utilizing Firebase, saving a limited amount of data offline is something Firebase is capable of doing, which can be sent to the cloud when the device has connected to the internet. Jonas will be responsible for establishing a connection between the EZ Tracker device to Google’s Firebase database. We will not have our Raspberry Pi connect to the Android phone as we had initially intended for the process is slightly too complicated for regular consumers and users. Jonas will also report if any additional information will be needed, created, and added to the database. The database will be implemented with constraints that will prohibit others users from reading and or access unauthorized user data. Other rules will be implemented as needed throughout the project.

### Hardware Specifications and Breakdown

The continual development of our Broadcom Development Platform, the Raspberry Pi, will be a combined effort amongst JRD Developers. In order to progress the project, the sensors must first be combined to a single entity that must be recognized on a sole system. JRD Developers will collaborate to create a new PCB on the open source software, Fritzing that will enable the Pi to work with multiple sensors on a single board. Jonas will be responsible for connecting and testing the hardware components. Delroy will assist Jonas in the SSD1306 OLED display that will read the values from LSM303 Accelerometer and Magnetometer. Jonas will be held accountable for the capacitive touch sensor which will allow the user to control the device. Functionality will include changing what’s being shown on the display, starting, stopping, and restarting the device. This will allow the user to see the data they want to see, especially due to the display being small in size. As the project nears the final stages, the device will have to be re-housed into a more robust, sleek, and compact design. Delroy will be in charge for the re-design and refinement of the case using Corel Draw, and working with the prototype lab for laser cutting. Jonas and Ryan will help with additional suggestions and ideas. The hardware needs to be suitable to be carried by a person over a lengthy period of time, whether by holding on to it, or strapping it onto themselves. Therefore, the device needs to be compact. Measurements and revisions will be made to allow the EZ Tracker to have a small of a footprint as possible.

# Build Instructions

## Budget and Parts

Firstly, a list of parts will be required in order to recreate this project. Below is price a breakdown of the prices and parts for each component used to get the project to completion. Spreadsheet with hyper links can be found at the [EZ Tracker Repository.](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/EZ_Parts_Budget.xlsx)

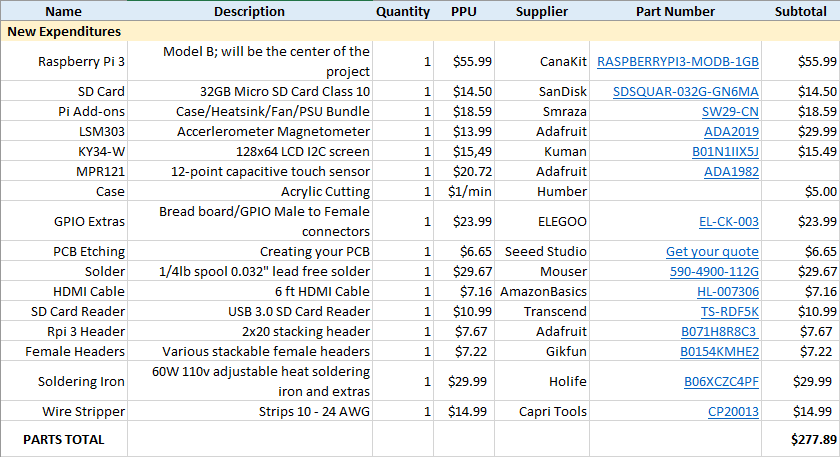


Figure 2

## Assembly

### Preparing the Pi

In this step we will cover basic Raspberry Pi imaging so you are able to login and access your device to test and drive your sensors. Download the latest [Raspbian image](https://www.raspberrypi.org/downloads/). Our recommendation is NOOBS. This will ensure that starting off, you will have almost everything you need if you decide to re-purpose the device later on. However, if you are comfortable with the command line and/or knowledgeable in using and navigating in Linux with commands, I suggest going with [Raspbian Lite](https://www.raspberrypi.org/downloads/raspbian/) because this will allow the device to not use as much power, and ensure the greatest compatibility with the Raspbery Pi. You are free to use other OSes like [Ubuntu MATE](https://ubuntu-mate.org/raspberry-pi/), [Ubuntu Server](http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/ubuntu-18.04.2-preinstalled-server-arm64+raspi3.img.xz), Arch Linux, etc. as long as it has systemd because we will be dealing with Python & systemd in this project There are might be some steps that will require different tools to do certain things, such as setting up static IPs using [ifupdown](https://manpages.debian.org/stretch/ifupdown/ifup.8.en.html) on Raspbian vs [Netplan](http://manpages.ubuntu.com/manpages/artful/man5/netplan.5.html) on Ubuntu. This guide will be focusing on Raspbian.

1. Download [etcher](https://www.balena.io/etcher/). This program will allow you to burn the Raspberry Pi image to your SD card.
2. Insert your SD card into the SD card reader and plug it into your computer.
3. Open etcher and follow the on screen instructions to burn your image. I found this program the easiest to use. [Extra documentation if needed](https://www.raspberrypi.org/documentation/installation/installing-images/README.md).
4. Insert the SD card into your Raspberry Pi along the underside of the device, logo facing out. Plug all of the required cables in such as: Ethernet, HDMI cable, mouse, keyboard, power, and turn the device on.
5. Upon boot you will see an option for different operating systems. Select **Raspbian** and follow the on screen instructions to complete the OS install.
6. At this point the Pi should boot to desktop. Follow the additional set up options on screen.
7. Make sure the Raspbian is up-to-date by running:   
    *$ sudo bash -c “apt update && apt upgrade”*
8. Enable I2C. This is the method we will be using to communicated with our Raspberry Pi. Type this command into the terminal:

*$ sudo apt-get install -y python-{smbus,pil} i2c-tools python3-pip apache2 git python{2,3}-dev build-essentials*

1. Type *sudo raspi-config* into the terminal.
2. Go to interfacing options > and enable I2C.
3. Install required Python packages by running:  
    *$ sudo pip3 install Adafruit-{MPR121,SSD130,LSM303} pyrebase*
4. Add this following entry to the sudoers file in order to use all commands without the need to type a password (we will use the user eztracker in this case).
   1. Edit the sudoers file:  
       *$ sudo visudo*
   2. Add this line at the end of the file  
       *eztracker ALL=(ALL) NOPASSWD:ALL*
5. We will be using a static IP address in order to access the HTML/PHP webpage, which will run the python file “eztracker.py”. Edit the “eth0” entry in **/etc/network/interfaces** to setup static address through ethernet. You can use GNU Nano, Vim, or any other text editor of your choice. Make sure that you are editing with root privilege
   1. Open the interface file in a text editor (We will use Nano for our examples):

$ sudo nano /etc/network/interfaces

* 1. Edit the “eth0” entry.  
     **Note: make sure to replace “dhcp” with “static”**:

iface eth0 inet static  
 address 169.254.2.13  
 netmask 255.255.0.0

* 1. Rest the networking using systemd:

*$ sudo systemctl restart networking*

1. Because EZTracker is aimed to be a portable device, it is best to have a smartphone connect to it wirelessly, such as WiFi direct. We will setup an access point on the Raspberry Pi for a phone to connect to. **Note: We never got WiFi direct to fully function. However, we hope that others who may attempt this will achieve success.**
   1. Enable systemd-networkd and systemd-resolved

$ sudo systemctl enable systemd-networkd  
$ sudo systemctl enable systemd-resolved

* 1. Create a file called “12-p2p-wlan0” and enter the following:  
     **Note: you can enter any IP address you wish.**

*[Match]  
Name=p2p-wlan0-\*  
  
[Network]  
Address=172.24.2.13/24  
DHCPServer=yes*

* 1. Before we set a custom SSID and password, we must allow wpa\_cli to autogenerate the entries for us, and change the credentials that need to be changed:

$ wpa\_cli -i p2p-dev-wlan0  
> p2p\_group\_add persistent  
> p2p\_group\_remove p2p-wlan-0  
> quit

* 1. Go to /etc/wpa\_supplicant/wpa\_supplicant.conf. In that file, you can change the the SSID and psk to your preferred SSID and password, respectively.

ctrl\_interface=DIR=/var/run/wpa\_supplicant GROUP=netdev  
 update\_config=1  
 device\_name=DIRECT-PB-RPi3  
 p2p\_go\_ht40=1  
 country=CA  
 network={  
 ssid="DIRECT-PB-RPi3"  
 bssid=56:1d:c5:95:c2:e9  
 psk="friedchicken"  
 proto=RSN  
 key\_mgmt=WPA-PSK  
 pairwise=CCMP  
 auth\_alg=OPEN  
 mode=3  
 disabled=2  
}

* 1. Test and see if it works

*$ wpa\_cli -i p2p-dev-wlan0*> *p2p\_group\_add persistent=0*

1. Power down the Pi

*$ poweroff*

### Breadboarding and prototyping

Here we will cover basic sensor connectivity to the Raspberry Pi using a breadboard for creating mock layout/design that will be used in the PCB creation stage.

1. Gather the following items: Breadboard, LSM303 Sensor, MPR121, 128x64   
   LCD sensor, and 4 Female-to-Male GPIO cables.
2. Identify the labels on the sensor. For basic usage, we will be using: 3.3v, GND, SDA, and SCL.
3. Identify the corresponding pinouts on the Raspberry Pi. [This website is a great tool to use if you are unsure.](https://pinout.xyz/)
4. Plug in the female part of the GPIO cables into the Raspberry Pi’s 1,3,5, and 6 pins (These are 3.3V, SDA, SCL, and Ground), based on the chart from pinout.xyz. These are the pins we are going to be using for this project.
5. Using the connectors that came with the sensors, plug them into the breadboard and rest the sensors accordingly. Below is an example.

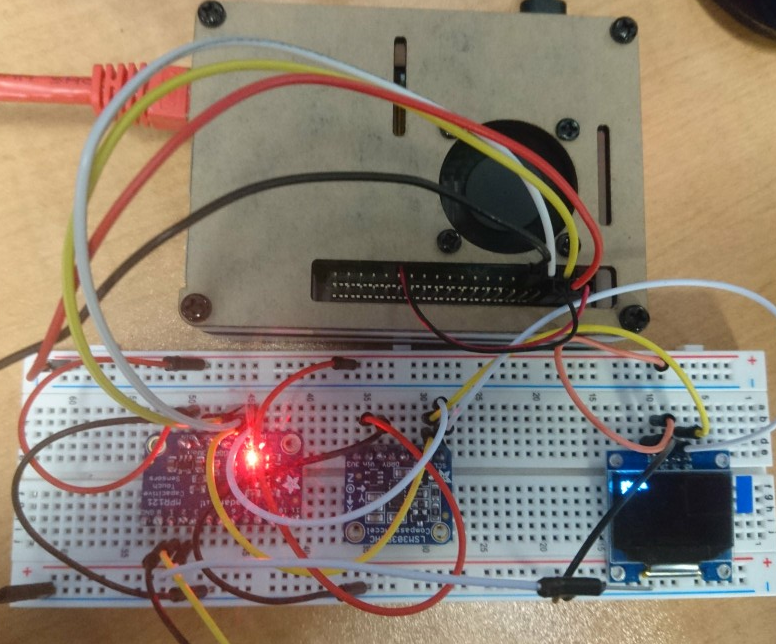


Figure 3

1. Connect the corresponding cables from the Raspberry Pi into the matched holes for the sensor.
2. (Optional but recommended) Power on the Raspberry Pi to see if the sensors are being detected correctly by viewing their address values.
3. (Optional but recommended) Open the terminal with *ctrl + alt + t* and run the command: *i2cdetect –y 1.* Ideally, the resulting output should be as follows, if not double check your connections and try again.



Figure 4

1. Voila! You have your mock up sensor connection!

### PCB Design

This is part of the project that needs to be proceeded with care and caution. It is advised to double check your designs before purchasing etching and cutting services. For this step, [Fritzing](http://fritzing.org/download/) will be used. It is an open-source application that allows the user to easily create PCB schematics for different development platforms. It is highly customizable and easy to use.

1. [Download](http://fritzing.org/download/) and extract Fritzing. Installation notes are on the linked page for various operating systems.
2. (Optional) Download the [AdaFruit Fritzing Library.](https://github.com/adafruit/Fritzing-Library) Handy if you want to take the extra step and create a mock or your own connection/designs in Fritzing.
3. Download our Fritzing file [here](https://github.com/YamiYukiSenpai/EZTracker/blob/master/pcb%20files/ez_v1.fzz) and open it. From the PCB tab, you can make changes at your leisure and pick it apart to see how it was made. Below is an image of our design.

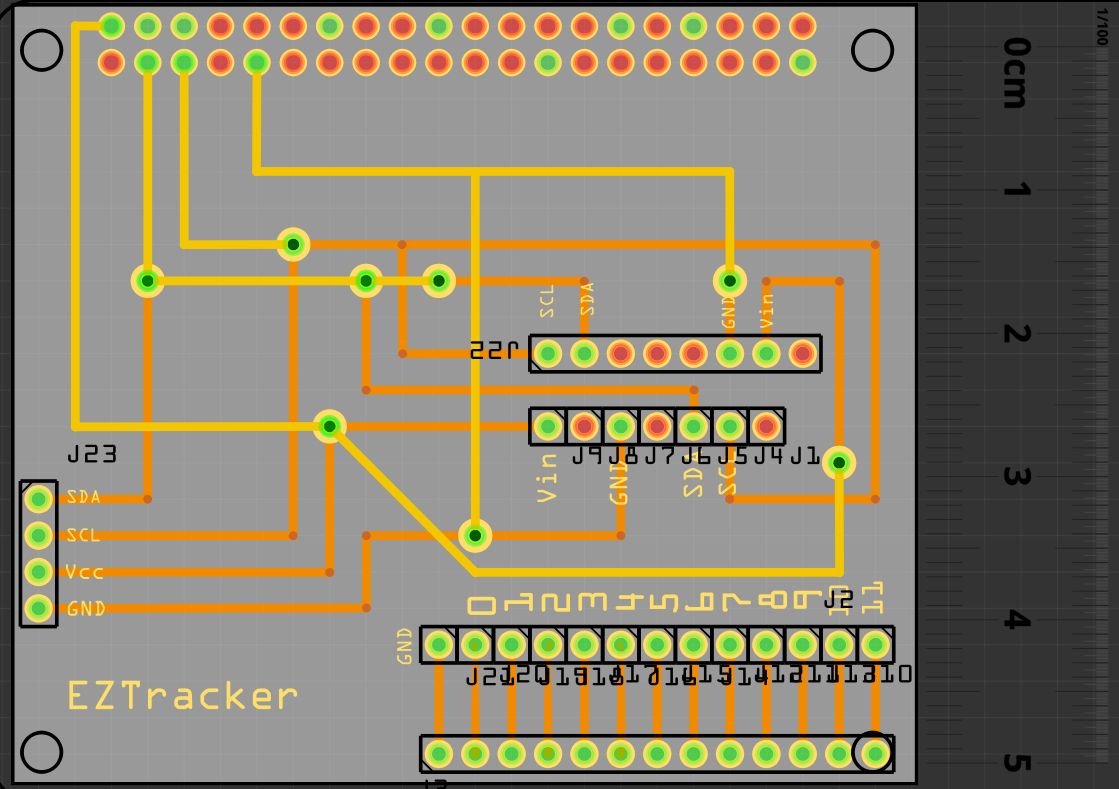


Figure 5

1. Export as a gerber file (the format primarily used in create a physical PCB). *File > Export for Production > Extended Gerber* and select an appropriate folder.
2. Zip/Compress the folder containing the gerber files and send them to your etcher of choice.

### Soldering

Once you have your PCB etched, we are ready to solder the parts together. Again, please double check your design before finally soldering.

1. Gather your sensors, copper wire, wire stripper, the pin headers that came with the sensor, your PCB, solder, and soldering iron.

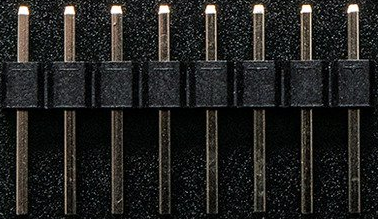


Figure 6

1. Solder your headers that came with your sensors, pictured above, to the corresponding holes. Put the longer end of the headers into your breadboard and place your sensor holes into the upright pins and solder all of the pins. This will ensure your sensor doesn’t move too much during soldering and a sturdy connection. **Note:** [Watch this video on soldering tips for additional help.](https://www.youtube.com/watch?v=oqV2xU1fee8) **Solder in a well ventilated area and use safety glasses.**



Figure 7

1. Solder your vias on your PCB. The easiest method I found, was to strip your copper wiring, stick it into the breadboard, and slide one via onto it so that it is flat and stable for soldering. Imaged below is an example, and repeat for each via. Once each via is soldered, snip the excess wire with cutters.

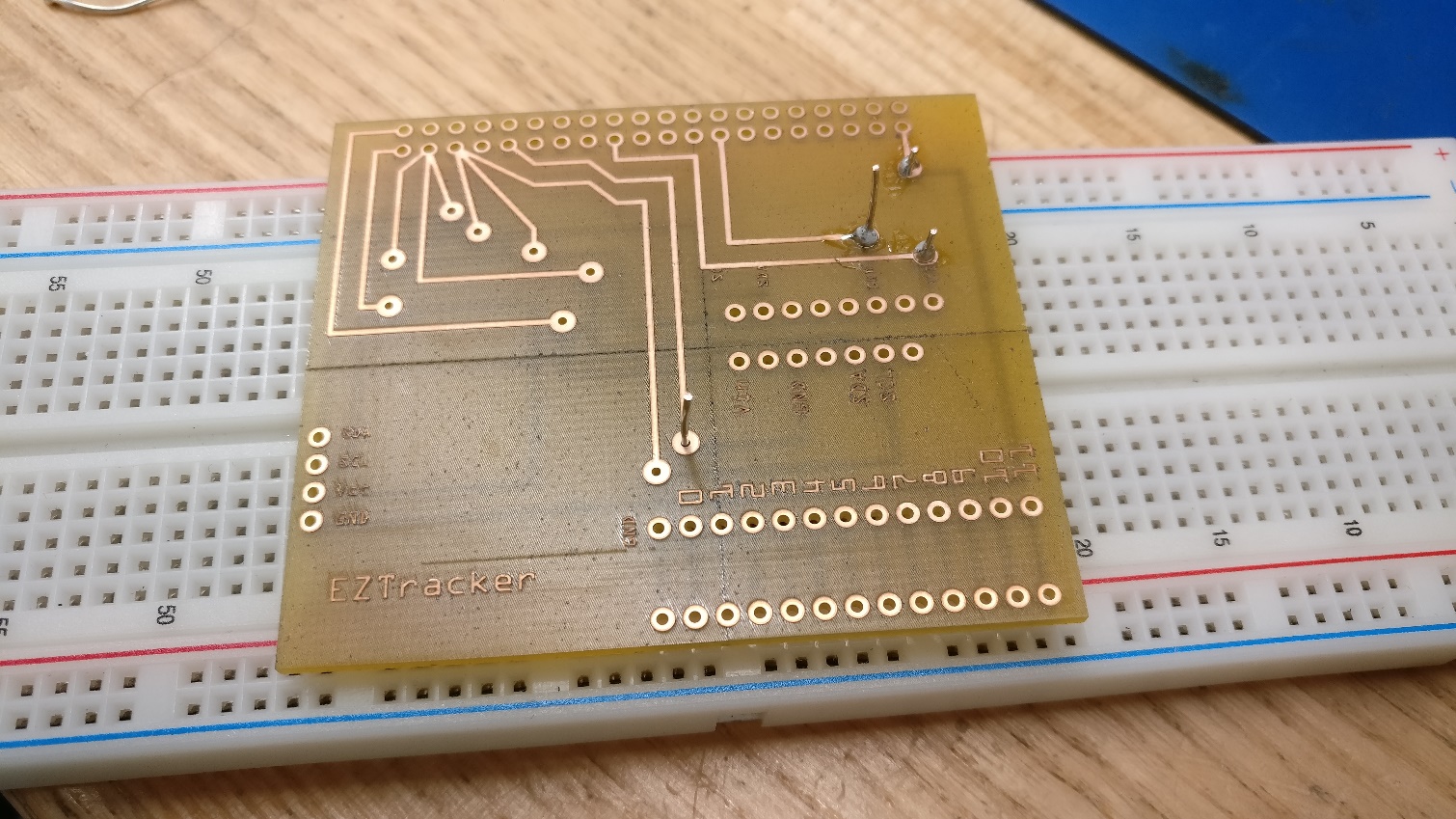


Figure 8

1. Solder your stackable headers. Using the breadboard again, take some extra copper wire from the snipped vias, or strip more, and place it into the female part of the header. The more, the sturdier. Flip the header over and plug it into the breadboard. Place your PCB onto the pins sticking out and solder the connections. An example is imaged below.

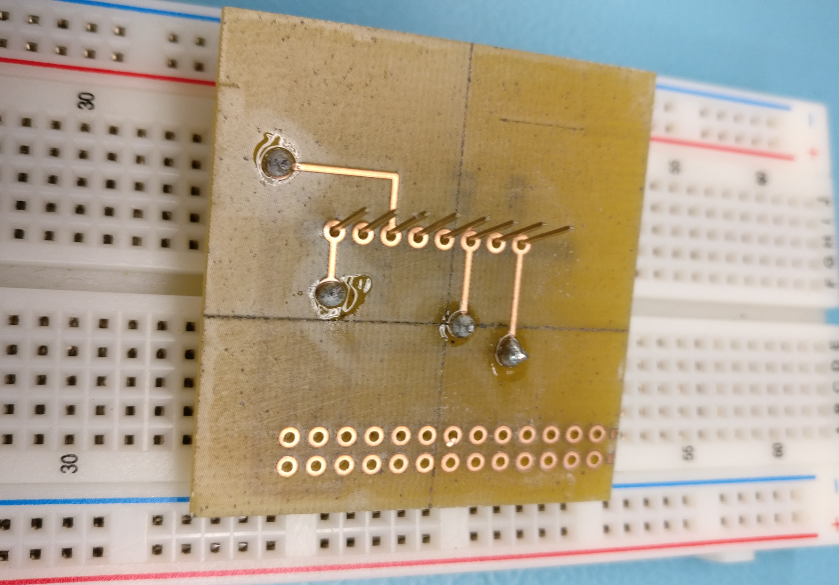


Figure 9

1. Repeat the process for the 2x20 header for the Raspberry Pi. It’s not entirely necessary to solder all of the holes, but it ensures the PCB does not bend as much.
2. Your soldering should be complete. Plug everything in accordingly, and you should be ready for the next step! Example placeholder image below.

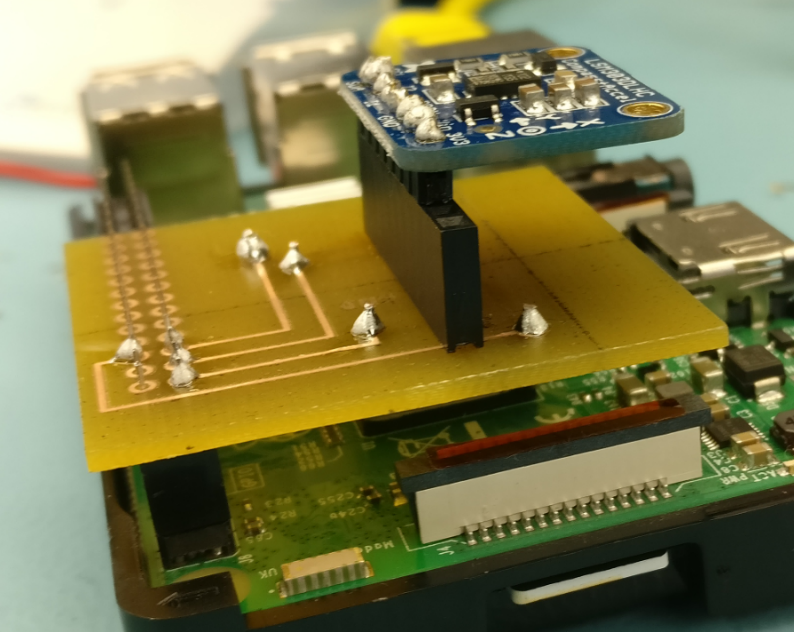


Figure 10

### **Power Up**

Once everything is plugged in, and you have double checked your connections, power on your Raspberry Pi.

1. Open up the terminal with *ctrl + alt + t* and run the command *i2cdetect – y 1*.
2. Hopefully everything is in working order. If so, you will see the following address values.



Figure 11

1. If not, go back and check if your sensors are connected accordingly.

### Unit Testing – LSM303

For this portion, you will need an internet connection as you will be required to download libraries in order to test each sensor.

1. Make sure your Pi is up to date with the latest packages. Run *sudo apt update* in the terminal to make sure everything is up to date.
2. Navigate to the */Adafruit\_Python\_LSM303/examples* directory.
3. Test your sensor by running *python simpletest.py*. Your readings should look like the following:

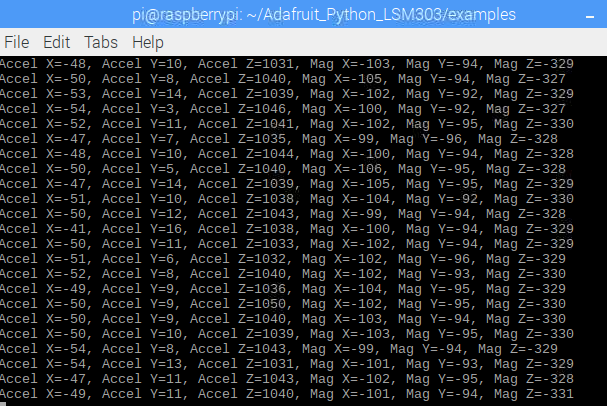


Figure 12

1. You can test the readings by moving your device in different directions with different speeds. You will notice the values changing accordingly.

### Unit Testing – MPR121

1. Download the test code from Jonas’s repository with:

*wget* [*https://raw.githubusercontent.com/YamiYukiSenpai/MacroKeyTouchSensor/master/cap-touch.py*](https://raw.githubusercontent.com/YamiYukiSenpai/MacroKeyTouchSensor/master/cap-touch.py).

1. Run the code:

*sudo python cap-touch.py*

1. You will then be able to touch each of the nodes on the sensor, or you can have individual wires running from each position. Below is an example output.

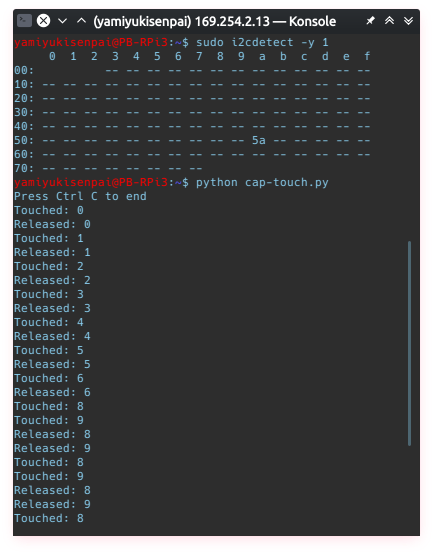


Figure 13

### Unit Testing – 128x64 Monochrome LED

1. Download the test code using *git clone https://github.com/adafruit/Adafruit\_Python\_SSD1306.git*
2. Navigate to the *Adafruit\_Python\_SSD1306/examples* directory and run the example code using *sudo python shapes.py*. The following should look like the placeholder image below.

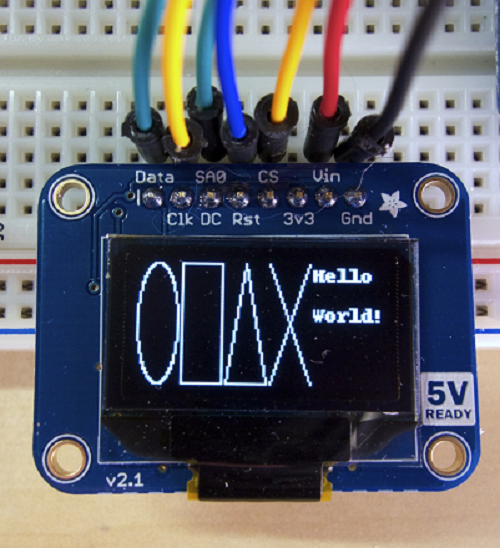


Figure 14

### Production Testing

[I have provided a case file](https://github.com/rfmaynard/Accel-MagnetoMeter/blob/master/case%20files/easeOmeter_CaseFiles_v4.cdr) (Created with CorelDraw x6) for those that wish to continue further for potential real world use. This case will ensure your sensors, Pi, and PCB are protected all while providing enough room for you to remove the sensor at your leisure. For portability, there is definite room that can be improved upon if you wish to make the device smaller and more portable. This concludes the end of the build instructions, and below is an image regarding the final product.

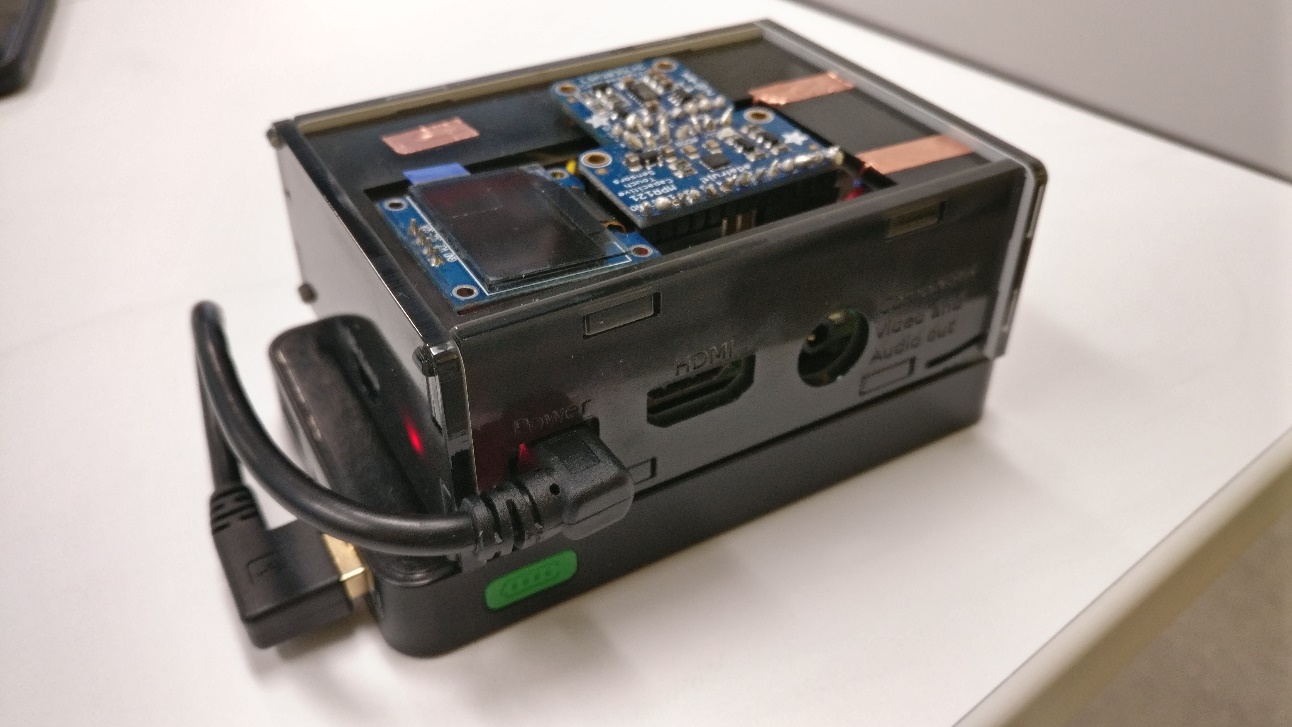


Figure 15

# Overall Description

## Product Functions

The application has five major functions. The functions include daily step counting, weekly step counting, calorie tracking, updating user information, and setting a weekly goal. Before diving into the EZ Tracker’s Android application, JRD Developers first needed to establish what data the application would use, and how to manipulate that data for EZ Tracker’s goal. We needed to centralize our criteria around the main function of the application, and it was determined to be the step and calorie tracking. With some research, we were able to find two key pieces of information that would be required from the user to ultimately calculate the approximate calories per step. The two metrics needed for the calculations are **weight** and **height**.

Firstly, the user’s weight can be used to determine the approximate calories burned per mile based on the constant 0.57. This is not an official constant but it was derived by reverse engineering a Fitbit (Banas, 2018).

Calories *Per Mile = 0.57 \* Weight (lbs)*

Secondly the user’s height can be used to roughly determine their stride length. Stride is the length of one step. By using the constant 0.414 (Sutton, 2016), multiplied by the user’s height in centimeters, we can roughly determine a user’s step length, or stride. We personally derived 0.414 from an average of the two constants 0.415 (Male) and 0.413 (Female), as we did not want to complicate users with a designated gender.

*Stride Length = 0.414 \* Height (cm)*

Thirdly, based on the variable above (stride length) we can now calculate approximately the amount of **steps a user will walk in a mile** by converting a mile to centimeters (160,934.4 cm) and dividing that number by their stride length (step length).

*Steps per Mile = 160,934.4 (cm) / Stride Length*

Lastly, based on the above variables Calories per Mile and Steps per Mile we can then determine the Calories per Steps by dividing the two respectively.

*Calories per Step = Calories per Mile / Steps per Mile*

Calories per Step can then be used to multiply into the user’s steps and thus determine their calories based on their weight, height and steps taken.

Afterwards, the matter of determining and putting the steps on to the database for the Android application to read and manipulate was the next focus. The LSM303 Accelerometer already came equipped in reading values from it’s X, Y and Z axis using the test code provided by Adafruit. What we needed to determine was how we could get a step from such readings. What Jonas decided to do was get all of the axis values every 0.5 seconds (which we determined to be the amount of time for 1 step through our own personal stepping), and compare the previous axis values to the new axis values. This method could then detect the changes in movement. By then looking for a value difference of about ±160 (based on our own tweaking and testing through walking with the device) in any of the axis values, the step counter would then increment by 1.

With the key points in place, we then could work with our previous semester’s database and Android application. To get an approximate overview of how the entire project tied together, we created a simple use case below.

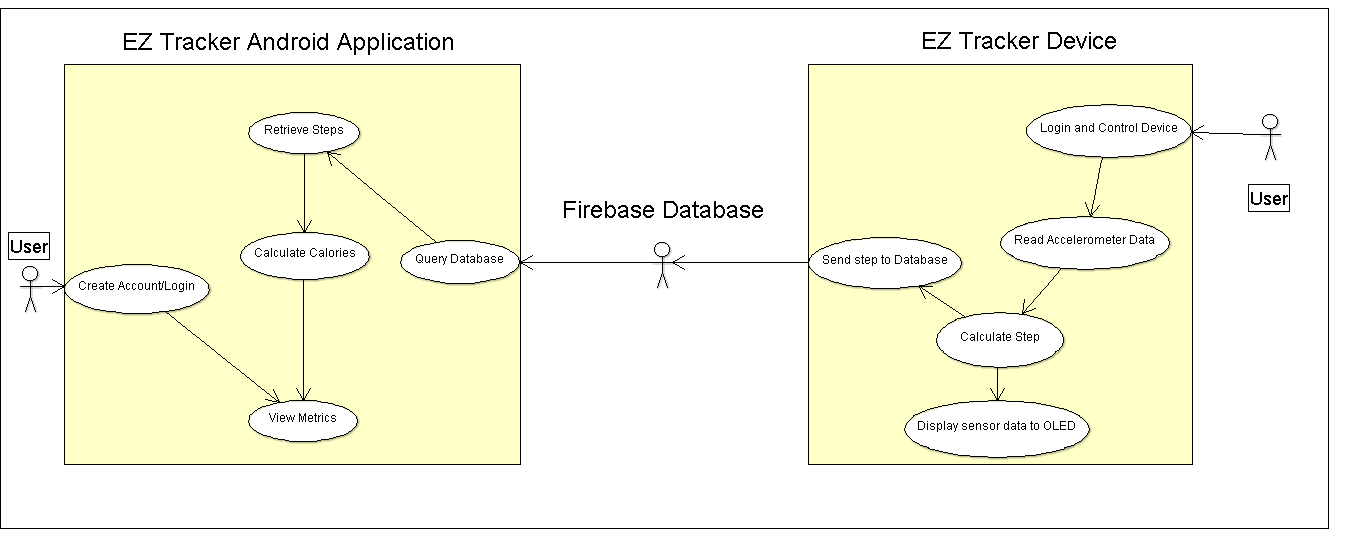


Figure 16

## Android Application Overview

The daily step counting will show the user their daily steps at a single glance. Below is an example of said function.

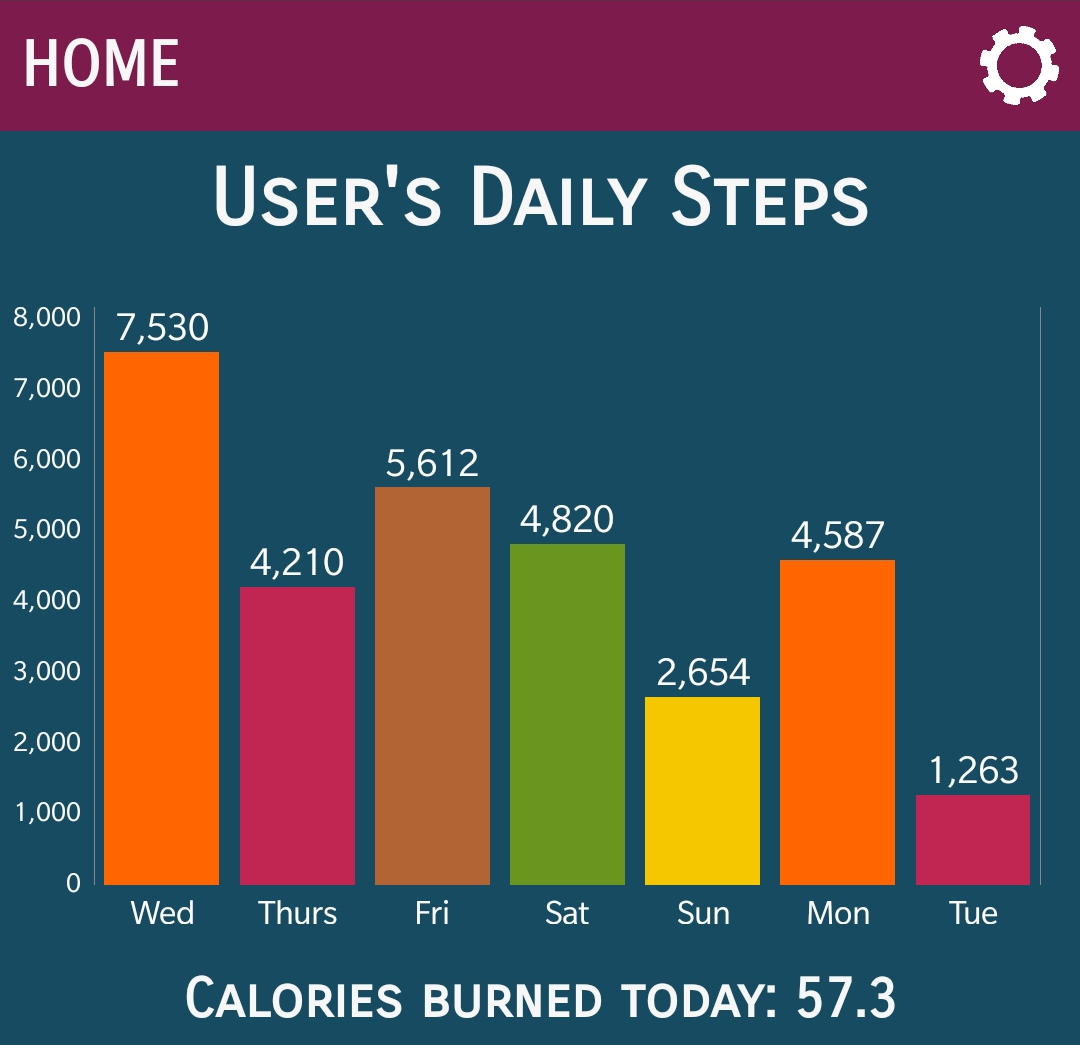


Figure 17

For example, the simple mode presents information in an easy-to-read text format with one graph or chart, depending on the information that the user selects. The advanced mode presents information in compact format with multiple graphs and charts. The step counting function is a feature that will count the number of steps a user has done. It can be turned on or off, depending on the activity (i.e, turned off for cycling)

* **Calorie tracking:** a function to calculate the calories burned based on distance traveled and steps taken.
* **Speed:** the feature where the speed at which the user traveled to accomplish his specific goal is calculated.
* **Goals:** users will be able to set goals for themselves for the week, and will be able to monitor said goal with ease in a simple, plain graphical format.

## User Classes and Characteristics

* Elderly
  + Simple interface allowing for easy use
  + Large font options for readability
  + Most important user class due to the rising elderly population and with health/fitness on the minds of most people.
* Power Users
  + Advanced interface available for the tech-savvy users.
  + Detailed statistics for the avid health tracking user.
* Cyclists
  + Robust hardware allows user to never have to take out his/her phone while cycling.
  + Can monitor basic data on the built-in display.

## Operating Environment

The software will operate in conjunction with a Broadcom development platform (Raspberry Pi) as the main engine, a capacitive touch sensor/effector, an accelerometer, and an OLED monochrome display; powered by the Android 5.0 and above operating system as the mobile client. On the back end, the system will be linked to Linux based web server and Google’s Firebase database for data storage and retrieval.

## Design and Implementation Constraints

The program will be created using the Java programming language with Android libraries. Any smartphone that is capable of running Android 5.0 Lollipop will be able to use this app.

The following constraints can pose a possible threat to the complete functionality of the system:

* Users API’s version may not be up-to-date to install the application
* Device may not have enough disk space for the installation (mobile app)
* Google play store may not be available in the user’s demographic location
* Other mobile constraints may prevent the installation of the app

## User Documentation

As the app is still in the development stages, all future documentation/instructions will be found at: <https://github.com/YamiYukiSenpai/EZ_Tracker/tree/master/documentation>

# Assumptions and Dependencies

While the system is fairly simple to use, the assumption can be made that the system may be affected due to certain situations. These include but are not limited to:

* Browser compatibility- the system might not be able to run an outdated browser. Thus, users a being recommended to have the most current version of the browser installed on their device.
* OS Compatibility- The system might not be able to operate on versions of Android that are older than Lollipop, which is deemed outdated by the developer.
* Internet connection not available.

For the system to be fully functional and effective, it will be dependent on the following:

* Internet Connection- wile internet access is not absolutely necessary for the system to function, it does require internet access at some point to interact and retrieve data from the database stored on a web server.

## Hardware Interfaces

The EZ Tracker app interacts with a few hardware components to accomplish its tasks. Firstly, system is powered by a Broadcom development platform (Raspberry Pi B+) as its main engine. It also uses a small 128x64 SSD1306 OLED Display for simple, on the fly, statistics. Tracking data is made possible by the LSM303 Accelerometer and Magnetometer, while the MPR121 12-Point Capacitive Touch Sensor offers seamless interaction between the system and the user. Data collected will be recorded to the Trackers local flash SD card storage, and the online database. This will be read to the small built-in screen as well as displayed into the EZ Tracker application. Supported Android devices include Android 5.0 and above.

## Software Interfaces

Connections for the EZ Tracker

* Database: Google’s Firebase will be used as the database. Used for storing tracking metrics such as direction, speed, steps, etc. Will only be an outbound connection.

Connections for the EZ Tracker application

* Database: Inbound communication from Firebase. Will display information into the application and display.
* Operating System: Internet Connection (HTTP) to send a request and receive e-mail notifications/instructions for password reset (SMTP).

## Communications Interfaces

EZ Tracker will employ communication through a number of interfaces. The first method being the database between the Tracker and the phone. The logs will be stored on the database from the Tracker and corresponding statistics from the database onto the app (via HTTP/Internet). Secondly, users will have their e-mails tied to their accounts on the app. This will allow email notifications to be received without storing any mobile number information (via SMTP). E-mail notifications can be disabled. A feedback form will also be available from the settings menu for users to contribute any feedback to the developers.

# System Features

## Simple Mode

### Description and Priority

This mode will be the main attraction point for the app. In general, users just want their product to work. It will display the simplest forms of data required to be considered a health tracker. At a glance, users will be able to see all of the information they need to.

Priority Level: High

Risk: 2

Cost: 2

## Stimulus/Response Sequences

Preconditions: The user is logged in and the device is connected to the tracker, and the app has not been changed to Advanced Mode.

1. User wishes to see their up to date tracker stats.
2. User opens EZ Tracker and the Simple Mode page is displayed (calories, steps, goals, etc.).
3. User is able to view the basic stats they want to see for their general health/activity.

# Functional Requirements

REQ-1: must download and install app from the Google Play Store.

REQ-2: Must be running android 5.0 with at least API 27 on mobile device

REQ-3: User must be registered and logged into the application

# Other Non-functional Requirements

# Performance Requirements

Upon opening the application on the device. The user will be able to see their up-to-date statistics within 5-10 seconds. The OLED display on the EZ Tracker will continuously update in real time since the numbers are local. In the case of querying the database, connection can depend on the users signal strength and will timeout if not successfully refreshed after 20 seconds.

# Safety Requirements

## Database corruption

The information in the database can be back up with the master-slave method in order for users to retain their data in case of data loss.

## Security Requirements

The mobility domain has a privacy sensitive nature, specifically with regards to the location tracking of users. In order to create a viable offering for the user we will build a simple, transparent system that can be understood and trusted by the people that are using it.

In order to build trust with the users of our system, the system shall make use of the following strategies:

* Anonymization & aggregation, so that information may be shared safely without disclosing personal information.
* Encryption, for all data that is privacy sensitive, but must be persisted on the server in order for basic functionality
* Open source / disclose security policies & practices
* Permit the use of anonymous avatars / aliases.
* Give control to end-users over private data (at least a delete private repository option)

# Software Quality Attributes

* **Usability:** checking that the system is easy to use and intuitive for the people not comfortable with current technology.
* **Maintainability:** any crashes regarding the app will prompt the user to send the crash report to the developers.

# Business Rules

Documents or other materials used for this project cannot be used for commercial purposes without the knowledge and consent of the developers

# Status Reports

## Status Report 1

The objective of this report is to provide details on the status of all team members working on the [EZ Tracker Internet of Things project](https://github.com/YamiYukiSenpai/EZTracker), and the current progress we have made so far.

**Hardware**

The hardware components of this project include the integration of three sensors with the [Broadcom development platform (Raspberry Pi)](https://www.amazon.ca/CanaKit-Raspberry-Starter-Premium-Black/dp/B07BCC8PK7/ref=sr_1_3_sspa?s=electronics&ie=UTF8&qid=1548973415&sr=1-3-spons&keywords=raspberry+pi+3+b&psc=1). The sensors we are using include a [LSM303 Accelerometer & Magnetometer](https://www.amazon.ca/Adafruit-Triple-axis-Accelerometer-Magnetometer-Compass/dp/B0149KK6SW/ref=sr_1_1?s=electronics&ie=UTF8&qid=1548973047&sr=8-1&keywords=LSM303+Accelerometer+%26+Magnetometer), a [SSD1306 Monochrome OLED](https://www.amazon.ca/yellow-128x64-SSD1306-display-module/dp/B017SZ847K), and a [MPR121 Capacitive Touch](https://www.amazon.ca/Adafruit-12-Key-Capacitive-Sensor-Breakout/dp/B00SK8PVNA/ref=sr_1_2?ie=UTF8&qid=1548973079&sr=8-2&keywords=MPR121+Capacitive+Touch) sensor.

**Acquisitions & Financial Status**

To date, we have acquired an extra LSM303 Accelerometer & Magnetometer, SSD1306 Monochrome OLED, and MPR121 Touch Sensor. The LSM303 by itself cost [$42](https://www.amazon.ca/Adafruit-Triple-axis-Accelerometer-Magnetometer-Compass/dp/B0149KK6SW/ref=sr_1_1?s=electronics&ie=UTF8&qid=1548973047&sr=8-1&keywords=LSM303+Accelerometer+%26+Magnetometer) including shipping and the SSD1306 was [$14.40](https://www.amazon.ca/yellow-128x64-SSD1306-display-module/dp/B017SZ847K). During production, we believe it is a wise decision to purchase extra parts as mistakes or breakages may occur which can result in a delayed or rushed project. Since we have decided to acquire extra hardware, this is added cost to our budget. Therefore, our financial status have changed marginally.

**Challenges**

Due to the hardware being portable by nature, we are aiming for all sensors to fit onto one PCB as we aim for a compact device. This is a bit challenging since the working space within the Broadcom development platform is limited. However, we have found an [online resource](https://www.instructables.com/id/Raspberry-PI-Multiple-I2c-Devices/) to assist in this regard. Jonas has begun trials using all of the sensors on one breadboard. Once the hardware can interact seamlessly, Jonas will design the PCB on [Fritzing](http://fritzing.org/home/), with added assistance from Delroy and Ryan when required. It might be a good idea to use a [Pi Zero](https://www.raspberrypi.org/products/raspberry-pi-zero/) for a smaller form factor. Our group can discuss this in the following weeks.

**Opportunities**

Delroy and Ryan are currently working on the website and revising the [Android application](https://github.com/rfmaynard/EZ_Tracker/tree/master/eztracker). They are drafting basic designs for it with the plan of having a consistent interface between the website and the application. Their main goal at this point is to read data from Google’s Firebase, and focus on design later. Having worked, and successfully interacted with the database on the Android app, the web based version should flow together with few issues.

Sincerely,

Team EZ Tracker

## Status Report 2

This report is to provide an update on the progress of the [EZ Tracker](https://github.com/YamiYukiSenpai/EZTracker) project to date. Its main focus is on the integration of the formerly independent hardware and software components into one cohesive unit.

**Hardware Progress**

As indicated in our last update, the aim is to integrate three sensors into one cohesive unit, operating on a single PCB. That hardware is also to integrate with an Android application working with an online database at the back end. At this point, we are still in the process of designing the PCB for the project, but hope to complete this task by the end of the week. This undertaking shouldn’t be that difficult as we have already integrated a working prototype of the sensors on a single breadboard. We are also in the process of designing the enclosure to be sent to the prototype lab for production. Based on our hardware design, this process requires very little or no modification to the enclosure design used for our individual sensors. We already have the android application working with the database, but now need to have the individual hardware read and write data to and from the database. Since we aim for a compact hardware design, one of the challenges we were facing was to get the sensors to work in a distributed fashion on the PCB, rather than a stackable one. However, our conceptual design overcame that challenge. Therefore, based on our current progress, the project is on track. As outlined in our work breakdown [schedule](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/EZ%20Tracker%20Status%20Report.pdf), and also within [budget](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/EZ%20Tracker%20Status%20Report.pdf). Hence, at this point, we have met all our objectives as defined in our [proposal](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/ProjectProposal_EZTrackerv1.pdf).

**Acquisitions & Financial Status**

No extra purchases have been required or incurred since the last progress report.

**Challenges**

A challenge we were having with the android app was to get the charts to display the dates correctly. We needed to determine how to manipulate the bar chart to get the current day of the week. This was required to display the current day on the right hand side of the graph. However, after much effort, we have resolved the problem and the charts are now presenting the inputted data correctly along with the correct day of the week. Regarding working with the sensors, we ran into issues trying to multi-process the sensors. We are in the middle of coding sensors working with one another and should have it completed within the next week.

Sincerely,

Team EZ Tracker

## Status Report 3

This report is to provide an update on the progress of the [EZ Tracker](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/EZ_Parts_Budget.xlsx) project to date. Its main focus is on the integration of the formerly independent hardware and software components into one cohesive unit.

**Hardware Progress**

As mentioned in the previous report, the [PCB](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/Pictures/pcb_solder.jpg) has been fully designed to take into consideration the integration of the [three sensors](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/Pictures/pcb_sensors.jpg). It was subsequently sent to the prototype lab for etching and has since been soldered and integrated into the Pi. However, the headers to accommodate the sensors are yet to be soldered. We are anticipating the design and hardware integration will be fully functional the first time around. To this end, Jonas is continuously working on integrating the sensors into a single working unit. Additionally, we are still contemplating migrating the project to a [Pi Zero](https://www.amazon.ca/Raspberry-Pi-v1-2-raspberry-zero/dp/B01L3IU6XS/ref=sr_1_8?hvadid=208253734008&hvdev=c&hvlocphy=9000980&hvnetw=g&hvpos=1t1&hvqmt=e&hvrand=2069421811900632789&hvtargid=kwd-298365187131&keywords=pi+zero&qid=1552598071&s=gateway&sr=8-8&tag=googcana-20). However, that consideration is dependent on the success of the PCB integration with the hardware sensors. That will make the migration to a new platform much easier. As it relates to the enclosure, Delroy is in the process of determining placement for a battery pack to power the device independently. Not to mention the added challenge of thinking about a design if we decide to downsize for the Pi Zero.

**Challenges**

The process of integrating the [software](https://github.com/YamiYukiSenpai/EZTracker/tree/master/Software/EZ_Tracker%20Beta%200.7) with the hardware is a bit tricky. Authenticating the Firebase portion is proving to be tedious, as the guide for Python-Firebase on [PyPi’s](https://pypi.org/) website did not work. After further research however, [Pyrebase](https://github.com/thisbejim/Pyrebase), a more up-to-date alternative, was the solution Jonas was able to utilize to successfully create an authenticated connection to our database. This portion of the work is proving to be the most time consuming part of the project. However, we are working assiduously to ensure the entire database completion and hardware integration by the end of next week.

**Financial Status and Remaining Work**

There has been no change to our current financial status as no new purchases have been made. Hence, our [budget](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/EZ_Parts_Budget.xlsx) has remained the same. However, that might change if we decide to migrate the project to the Pi Zero platform. So as it stands, as indicated in our [project breakdown schedule](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/EZ_Tracker_Schedule.PNG), we are currently on track and within budget. By the end of next week, we hope to have the entire project fully integrated and working together as one cohesive unit.

Sincerely,

EZ Tracker Team

## Status Report 4

This report is to provide an update on the progress of the [EZ Tracker](https://github.com/YamiYukiSenpai/EZTracker) project to date. Its main focus is on the integration of the formerly independent hardware and software components into one cohesive unit.

**Overall Progress and Troubleshooting**

As mentioned in the previous report, the PCB had been fully designed. Unfortunately, it did not work as we had intended. Our PCB would detect the sensors correctly, but when running our program it would not work correctly and errors were thrown we had not seen before. We had determined it to be faulty wiring as the project would succeed on a breadboard. We reduced the number of Pi GPIO pins (as indicated between [v1](https://github.com/YamiYukiSenpai/EZTracker/blob/master/pcb%20files/ez_v1.png) and [v2](https://github.com/YamiYukiSenpai/EZTracker/blob/master/pcb%20files/ez_v2.png)) we were connected to and branched our vias to accept inputs from the 3 sensors at once, essentially simplifying the overall design.

In addition, we had [our case](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/Pi2CaseEdit.pdf) designed by Delroy as per our group discussions. We had proposed a compartment below to house the rechargeable battery, and a port to charge and power our device. After receiving the [completed product](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/Pictures/caseResult2.png), we realized we had measured [incorrectly for the battery](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/Pictures/caseResult1.png), charging, and power source compartment. We have re-measured and Delroy will be in the process of re-designing in Corel Draw over the weekend, to be ready for next week’s milestone.

Lastly, Jonas is in the process of working with the Raspberry Pi’s WiFi direct system. This will allow the user to access an apache website hosted on the device, from their phone. This makes it easier for logging in and running the program from the mobile device. Jonas is still in the process of configuring this system and managed to get it working. Unfortunately, the WiFi direct had spontaneously stopped co-operating and is trying to figure out reasons and solutions to correct it.

**Financial Status and Remaining Work**

There has been no change in our financial status and our budget remains the same. Given the amount of remaining work left, we will most likely not be able to make the switch to a Pi Zero. We realize that further improvements can be made as revisions occur just like in any other product. We have opted to have a well working device the first time around rather than something that is smaller. By the end of next week we hope to have a functional standalone unit, the only thing left after that would be tweaking of the sensors for accuracy, and WiFi direct configuration. We also plan on completing all of the requirements of our Android application as well, which includes caloric calculations and launching the webpage to log into the Pi.

Sincerely,

EZ Tracker Team

## Status Report 5

This report is to provide an update on the progress of the [EZ Tracker](https://github.com/YamiYukiSenpai/EZTracker) project to date. Its main focus is on the integration of the formerly independent hardware and software components into one cohesive unit.

**Overall Progress**

As indicated in the previous report, we needed to complete sensor accuracy, final enclosure, Wi-Fi direct, and final Android application tweaking. We are proud to announce that we have hit majority of these milestones, which puts us very close to completion. The only task remaining is to solve Wi-Fi direct. The last week has been a decent sprint, and fortunately we have not run into any major roadblocks. We were able to test a standalone product where a user was able to carry around, tracks steps, and send data to the Firebase database, which is in turn was manipulated by the completed Android application.

**Challenges and Opportunities**

In the end, the project enclosure needed a total of four revisions. The first two were scrapped in terms of minute measurement discrepancies. The third version of the enclosure worked well, but due to our battery solution shorting, that revision needed to be scrapped and our backup plan needed to be put into effect. This version involved the battery pack being modular to the device as opposed to being directly part of it. Thanks to the first three versions, the [final revision](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Case%20Files/Pi2CaseRev4_clean.pdf) of the case was more precise.

Connecting the phone to the device through Wi-Fi direct was unfortunately not successful. Connecting to the login page using a static IP via Wi-Fi direct does not work as initially intended. However, the device can establish a standard internet connection and send the desired values to our Firebase database. Continual effort will be put into refining our sensors and Wi-Fi direct to ensure a seamless user experience. If a Wi-Fi direct connection cannot be established by the end of this week, the Pi’s Ethernet port will be used to access the login interface.

**Financial Status**

This week our team needed to make two small purchases. With the shorting of our previous battery solution, it was required to find another portable battery pack that is able to power our device. Along with that, we also needed to obtain a short micro USB cable to run from the battery pack to the power port on the development platform. Both purchases will be reflected in our [budget sheet](https://github.com/YamiYukiSenpai/EZTracker/blob/master/Documentation/EZ_Parts_Budget.xlsx) located on our repository.

Sincerely,

EZ Tracker Team

.

# Conclusions

The conception of the EZ Tracker IoT project was developed to address a special existing need in the market, and particularly for a specific demographic. During the course of fifteen weeks, we have done extensive research, development, and unit testing to ensure the product is ready and suitable for refinement, production, and distribution. We believe that with the integration of these technologies and the careful collaboration among team members, we have achieved our objective of producing a robust, inexpensive, and simple to use product for our target demography.

# Recommendations

As of now, due to sensor size constraints and time limitations, in addition to working with the full sized Raspberry Pi Model B, our product isn’t at the ideal size. In order to aim for a more wearable design and/or more compact profile, future editions may utilize the Pi Zero model with direct soldering of the sensors to the printed circuit board, rather than using the stackable headers to keep sensors modular. This would drastically cut down on footprint size as well as total weight. Furthermore, during the Capstone EXPO it was brought to our attention that another alternative to a Pi Zero could have been an Arduino. Regarding sensor usage, an accelerometer will do the job for this project. However, it won’t be as accurate in determining steps compared to a GPS, which could be implemented down the line to replace the LSM303 accelerometer.

As far as interfacing capabilities can go, another feature that could be implemented is the idea of logging into the EZ Tracker device via the Android application. Compared to the current method of using a computer to access the Web UI, this would make the overall interaction between the devices more streamlined. This would be achieved via Wi-Fi Direct.

Over time, further modifications can be added to the Android application. Ideally, this is a system catered to individuals lacking technological skills. However, as users become more adept in using EZ Tracker, perhaps the ability to view advanced metrics and settings can be enabled within the Android Application.

The following recommendations are made with respect to the use of the app and corresponding hardware for the best results:

- consider commercialization (e.g. What if you had to make a thousand of your IoT device? What is keeping your device from being used for its intended purpose today?).

# Bibliography

Banas, T. (2018, June 14). How to Convert Pedometer Steps to Calories. Retrieved April 16, 2019, from https://www.livestrong.com/article/238020-how-to-convert-pedometer-steps-to-calories/

Sutton, A. (2016, November 09). How to Determine Stride for a Pedometer by Height & Weight. Retrieved April 16, 2019, from https://livehealthy.chron.com/determine-stride-pedometer-height-weight-4518.html

YamiYukiSenpaiYamiYukiSenpai. (n.d.). How to get Raspberry Pi 3 & Android device to communicate with each other, and authenticate? Retrieved from https://raspberrypi.stackexchange.com/a/95517/68451

# Appendices

## Appendix A: Glossary

**APP –** Short form for application

**Database –** An organized collection of data, stored and accessed electronically.

**Android** – an operating system designed for mobile devices (i.e. cell phones, tablet computers) by Google, Inc.

**Android device** – any device (phone/tablet) running an Android Operating System. In this document, synonymous to “smart phone running Android.”

**Operating System** – the software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.

**SSD1306 Monochrome OLED Display** – a small display, about 1" diagonal, but very readable due to its high contrast. This display is made of 128x64 individual white OLED pixels, each one is turned on or off by the controller chip.

**LSM303 Accelerometer and Magnetometer** – a small chip that can determine axis movements as well as direction (North, South, East, West)

**MPR121 12-Point Capacitive Touch Sensor** – a touch sensor that can handle up to 12 individual touch points and can be implemented with nearly any microcontroller.

**SD Card** – Secure Digital card is a type of memory card typically used in digital cameras and other portable devices.

**HTTP** – Hypertext Transfer Protocol is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands

**SMTP** – Simple Mail Transfer Protocol is an Internet standard for electronic mail (email) transmission.

**Firebase** - a mobile and web application development platform developed by Firebase, Inc., then acquired by Google. This online platform stores for our application.

**systemd** - software for managing the init and services of the software